The APACHE ANT Project

“Another Neat Tool” (un autre chouette outil)
`ant.apache.org`

Nicolas Hernandez

IUT de Nantes – Département Informatique
LINA - Laboratoire d’Informatique de Nantes Atlantique

Cours de Licence Professionnelle
2007 – 2008

Nantes, le 22 novembre 2007

Sommaire

1. Motivations, principe de l'outil `ant`, installation, exécution
2. Structure du fichier de configuration `build.xml`
3. Les catégories de tâches
4. Mises en oeuvres des tâches classiques investies dans des cibles habituelles (compilation, génération de documentation, test et versionning)
5. Définir ses propres tâches
6. ant dans Eclipse
7. Sujet du TP

Introduction – Sommaire

Motivations

"L'objectif du projet `ant.apache.org` *est de fournir un outil pour permettre "*

- la **construction d'applications**
- l'**automatisation les opérations répétitives** du cycle du développement (nettoyage du projet, compilation, génération de la documentation, versionning, test, déploiement...)
- l'**indépendance envers toute plate-forme** (écrit en Java)
- la **configuration** à l'aide d'un fichier de XML qui décrit les tâches à exécuter
- l'**extension** en permettant l'écriture de nouvelles tâches

Principe

- la commande `ant` repose sur un fichier de configuration `build.xml`
- le `build.xml` contient un ensemble de cibles (`target`), qui constituent les étapes du projet de construction
- chaque cible contient une ou plusieurs tâches (`task`) ordonnées, qui constituent des traitements unitaires à réaliser
- chaque cible peut avoir une dépendance (`depends`) envers une ou plusieurs autres cibles pour pouvoir s'exécuter

`build.xml`

Fichier de configuration Ant
- Cible
  - Tache
- Cible
  - Tache
  - Tache

Installation

Download

`ant.apache.org`

Requirement

un JDK plutôt qu'un JRE (sans quoi des tâches indisponibles)

Setup (Linux/Unix bash)

```
export ANT_HOME=/mon/local/ant
export JAVA_HOME=/usr/local/jdk-1.5.0.05
export PATH=${PATH}:${ANT_HOME}/bin
```

Install dans le `ANT_HOME` *(Linux/Unix bash)*

`sh build.sh install`

Dependencies dans le `CLASS_PATH`

Entre autres : `xalan.jar` (XSL transformer), `junit.jar`, `mail.jar`, `Groovy jars` (scripts Java), `jdepend.jar` ...

Exécution

Utilisation en ligne de commande selon la syntaxe

`ant [options] [cible]`

Comportement par défaut

- recherche un fichier `build.xml` dans le répertoire courant
- si aucune cible n'est spécifiée, il prendra celle déclarée par défault dans le fichier

`ant`

Spécification d'un fichier de configuration

`ant -buildfile monbuild.xml`

Exécution de la cible `clean` et toutes les cibles dont elle dépend

`ant clean`

ant interfacé dans de nombreux IDEs

Existence de plugins pour Eclipse, NetBeans, IntelliJ IDEA...

Le fichier `build.xml` *– Sommaire*

Le fichier `build.xml` *et l’élément racine* `project`

Le fichier `build.xml` contient la description du processus de construction de l’application

Le prologue

`<?xml version="1.0" encoding="UTF-8">` (ou bien `"ISO-8859-1"`)

L’élément racine du document et ses attributs

- `name` : nom du projet
- `default` : cible par défaut à exécuter si aucune cible précisée
- `basedir` : répertoire de référence pour la localisation relative des autres répertoires

```
<project name="mon projet" default="compile" basedir=".">
```

Note concernant l’adressage de fichiers

- Quelque soit la plate-forme, un chemin utilise la caractère slash ’`/`’ comme séparateur
- L’expression `**/` permet de désigner tous les sous répertoires du répertoire défini dans l’attribut dir répertoire

Le corps du fichier `build.xml` et les commentaires

Les commentaires

```
<!-- Ceci est un exemple de commentaire -->
```

Le "corps" et les définitions des

1. **propriétés** (`properties`) : variables qui contiennent des valeurs utilisables par des cibles ou tâches
2. **ensembles de fichiers** (`fileset, patternset, filelist`) définis en spécifiant explicitement des répertoires, des patrons, des listes exhaustives de fichiers
3. **cibles** (`targets`), étapes du projet de construction qui mettent en oeuvre des **tâches**, traitements unitaires

(habituellement présentées dans cet ordre)

Les propriétés (définition)

Utilité

définir une seule fois une valeur qui est utilisée plusieurs fois dans le projet

Définition des variables

- avec l'option `-D nom=valeur` en ligne de commande
- avec la balise `property` dans le build.xml

Dans le build.xml

```
<property file="mesproprieteslocales.properties" />
<property name="projet.nom" value="mon_projet" />
<property name="projet.version" value="0.0.10" />
<property name="projet.license" location="doc/LICENSE" />
<property name="src.dir" value="src" />
<property name="build.dir" value="build" />
```

`file`, un fichier qui contient une liste de lignes `nom=valeur`
`location`, un fichier dont le contenu désigne une valeur

Les propriétés (utilisation)

Utilisation à l'aide de
`${projet.name}`

Ordre de définition des propriétés
seule la première définition d'une propriété compte, les suivantes sont ignorées

Propriétés prédéfinies (attention changements depuis ant-1.7)

- `basedir`, chemin absolu du répertoire de travail ;
- `ant.file`, chemin absolu du fichier build en cours de traitement ;
- `ant.java.version`, version de la JVM qui exécute ant ;
- `ant.project.name`, nom du projet en cours d'utilisation

Les ensembles de fichiers

Les ensembles de fichiers, `fileset`*, utilisés au sein d'une autre tâche*

- *dir* Répertoire de départ de l'ensemble de fichiers
- *includes* Liste des fichiers à inclure
- *excludes* Liste des fichiers à exclure

```
<fileset dir="src" includes="**/*.java">
```

Les ensembles de motifs, `patternset`

- *id* Identifiant pour l'ensemble qui pourra ainsi être réutilisé
- *refid* Demande la réutilisation d'un ensemble dont l'identifiant est fourni comme valeur

(ainsi que `includes` et `excludes`)

```
<fileset dir="src">
  <patternset id="source_code">
    <include name="**/*.java"/>
    <exclude name="**/*~"/>
  </patternset>
</fileset>
```

Les ensembles de fichiers

Listes de fichiers finies

- *id* Identifiant pour l'ensemble qui pourra ainsi être réutilisé
- *dir* Répertoire de départ de l'ensemble de fichiers
- *files* Liste des fichiers séparés par une virgule
- *refid* Demande la réutilisation d'un ensemble dont l'identifiant est fourni comme valeur

```
<filelist dir="texte" files="fichier1.txt,fichier2.txt" />
```

Liste traditionnelle de fichiers à exclure

```
**/*~
**/#*#
**/.#*
**/%*%
**/._*
**/CVS
**/CVS/**
**/.cvsignore
**/SCCS
**/SCCS/**
**/vssver.scc
**/.svn
**/.svn/**
**/.DS_Store
```

Les éléments de chemins, les tâches et les cibles

Ajout d'éléments à la variable classpath, pathelement

```
<classpath>
  <pathelement location="lib/mabib.jar">
  <pathelement location="ext/">
</classpath>
```

Les cibles <target>

ensemble de tâches à réaliser dans l'ordre de présentation

- *name* : le nom de la cible. obligatoire
- *description* : brève description de la cible. optionnel (utile pour les IDE)
- *depends* : liste des cibles dont dépend la cible. optionnel
- *if* : conditionne l'exécution par l'existence d'une propriété. optionnel
- *unless* : conditionne l'exécution par l'inexistance de la

Catégories de tâches – Sommaire

File / Directory / Archive tasks
- File / Directory tasks
- Archive Tasks

Development tasks
- Compile Tasks
- Documentation, Logging and Testing Tasks

Execution Tasks
- Execution Tasks
- Remote Tasks

Misc and Property tasks
- Misc tasks
- Property Tasks

Un aperçu complet des tâches standardes disponibles :
`http://ant.apache.org/manual/tasksoverview.html`

File / Directory tasks

File / Directory tasks

Copy Copies a file or Fileset to a new file or directory.

Delete Deletes either a single file, all files and sub-directories in a specified directory, or a set of files specified by one or more FileSets.

Mkdir Creates a directory. Non-existent parent directories are created, when necessary.

Move Moves a file to a new file or directory, or a set(s) of file(s) to a new directory.

Get Gets a file from a URL.

Patch Applies a "diff" file to originals.

... FixCRLF, Replace, ReplaceRegExp, Sync, Tempfile, Touch, Checksum, Chgrp, Chmod, Chown, Concat

Archive Tasks

Zip / Unzip Zips a set of files. / Expands a Zip file.

Jar/Unjar Jars a set of files./ Unzips a jarfile.

Manifest Creates a manifest file.

Rpm Invokes the rpm executable to build a Linux installation file. This task currently only works on Linux or other Unix platforms with RPM support.

Tar/Untar Creates a tar archive / Untars a tarfile.

... BUnzip2, BZip2, Cab, Ear, GZip, GUnzip War, Unwar

Compile Tasks

Compile Tasks

Javac Compiles the specified source file(s) within the running (Ant) VM, or in another VM if the fork attribute is specified.

Depend Determines which classfiles are out-of-date with respect to their source, removing the classfiles of any other classes that depend on the out-of-date classes, forcing the re-compile of the removed classfiles. Typically used in conjunction with the Javac task.

JspC Runs the JSP compiler. It can be used to precompile JSP pages for fast initial invocation of JSP pages, deployment on a server without the full JDK installed, or simply to syntax-check the pages without deploying them. The Javac task can be used to compile the generated Java source. (For Weblogic JSP compiles, see the Wljspc task.)

Documentation, Logging and Testing Tasks

Documentation Tasks

Javadoc Generates code documentation using the javadoc tool.

Logging Tasks

Record Runs a listener that records the logging output of the build-process events to a file. Several recorders can exist at the same time. Each recorder is associated with a file.

Testing Tasks

Junit Runs tests from the Junit testing framework. This task has been tested with JUnit 3.0 up to JUnit 3.7 ; it won't work with versions prior to JUnit 3.0.

JunitReport Merges the individual XML files generated by the Junit task and applies a stylesheet on the resulting merged document to provide a browsable report of the

Execution Tasks

Execution Tasks

Ant Runs Ant on a supplied buildfile,

AntCall Runs another target within the same buildfile,

Apply/ExecOn ; Exec Executes a system command.

Java Executes a Java class within the running (Ant) VM, or in another VM if the fork attribute is specified.

Parallel A container task that can contain other Ant tasks. Each nested task specified within the <parallel> tag will be executed in its own thread.

Sequential A container task that can contain other Ant tasks. The nested tasks are simply executed in sequence. Its primary use is to support the sequential execution of a subset of tasks within the <parallel> tag.

Sleep A task for suspending execution for a specified period of time. Useful when a build or deployment process requires an interval between tasks.

Remote Tasks

FTP Implements a basic FTP client that can send, receive, list, and delete files, and create directories.

Scp Copy files to or from a remote server using SSH.

setproxy Sets Java's web proxy properties, so that tasks and code run in the same JVM can have through-the-firewall access to remote web sites.

Sshexec Execute a command on a remote server using SSH.

Telnet Task to automate a remote telnet session. This task uses nested <read> and <write> tags to indicate strings to wait for and specify text to send.

Misc tasks

Mail A task to send SMTP email

Echo Echoes text to System.out or to a file.

Fail Exits the current build by throwing a BuildException, optionally printing additional information.

Input Allows user interaction during the build process by displaying a message and reading a line of input from the console.

Sound Plays a sound file at the end of the build, according to whether the build failed or succeeded.

Sql Executes a series of SQL statements via JDBC to a database. Statements can either be read in from a text file using the src attribute, or from between the enclosing SQL tags.

TStamp Sets the DSTAMP, TSTAMP, and TODAY properties in the current project, based on the current date and time.

XmlValidate Checks that XML files are valid (or only well-formed). This task uses the XML parser that is currently used by Ant by default, but any SAX1/2 parser can be specified,

Property Tasks

Property Tasks

Available Sets a property if a specified file, directory, class in the classpath, or JVM system resource is available at runtime.

Basename Sets a property to the last element of a specified path.

Dirname Sets a property to the value of the specified file up to, but not including, the last path element.

Condition Sets a property if a certain condition holds true ; this is a generalization of Available and Uptodate.

XmlProperty Loads property values from a well-formed XML file.

... Whichresource, Echoproperties, LoadFile, LoadProperties, MakeURL, PathConvert, Property, PropertyFile, Uptodate,

Exemples de mise en oeuvre de tâches – Sommaire

Tâche echo

`<echo>` permet d'écrire dans un fichier ou d'afficher un message durant l'exécution des traitements

message the message to echo. Optional

file the file to write the message to. Optional

append Append to an existing file (or open a new file / overwrite an existing file) ? Optional - default is false.

level Control the level at which this message is reported. Optional of "error", "warning" (-quiet, -q), "info" (no statement), "verbose" (-verbose, -v), "debug" (-debug, -d) (decreasing order) Optional - default is "warning".

encoding encoding to use, default is "" ; the local system encoding. since Ant 1.7 Optional

`http://ant.apache.org/manual/CoreTasks/echo.html`

Tâche echo

build.echo.xml

```
 <?xml version="1.0" encoding="UTF-8"?>
<project name="Test echo avec Ant" default="init" basedir=".">
  <!-- =================================================================== -->
  <!-- Initialisation -->
  <!-- =================================================================== -->
  <target name="init">
    <echo message="Debut des traitements" />
    <echo>
      Fin des traitements du projet ${ant.project.name}
    </echo>
    <echo message="Ceci est un message warning" level="warning" />
    <echo message="Ceci est un message debug" level="debug"/>
    <echo file="${basedir}/log.txt" append="false" message="Debut Traitement"/>
    <echo file="${basedir}/log.txt" append="true" >
Fin Traitement
    </echo>
  </target>
</project>
```

Tâche echo

ant -quiet -buildfile build.echo.xml

```
Buildfile: build.echo.xml

init:
     [echo] Debut des traitements
     [echo]
     [echo]       Fin des traitements du projet Test echo avec Ant
     [echo]
     [echo] Ceci est un message warning

BUILD SUCCESSFUL
Total time: 0 seconds
```

La tâche tstamp

<tstamp> définit trois propriétés :

- *DSTAMP* : la date du jour au format AAAMMJJ
- *TSTAMP* : l'heure actuelle sous la forme HHMM
- *TODAY* : la date du jour au format long

build.tstamp.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test tstamp avec Ant" default="init" basedir=".">
  <target name="init">
    <tstamp/>
    <echo message="Nous sommes le ${TODAY}" />
    <echo message="DSTAMP = ${DSTAMP}" />
    <echo message="TSTAMP = ${TSTAMP}" />
  </target>
</project>
```

ant -buildfile build.tstamp.xml

```
Buildfile: build.tstamp.xml
init:
     [echo] Nous sommes le November 21 2007
     [echo] DSTAMP = 20071121
     [echo] TSTAMP = 1120
BUILD SUCCESSFUL
Total time: 0 seconds
```

La tâche mkdir

<mkdir> Creates a directory. Also non-existent parent directories are created, when necessary. Does nothing if the directory already exist.
http://ant.apache.org/manual/CoreTasks/mkdir.html

build.mkdir.xml

```
 <?xml version="1.0" encoding="UTF-8"?>
<project name="Test mkdir avec Ant" default="init" basedir="."> \pause

  <target name="init">
    <mkdir dir="${basedir}/build" />
    <mkdir dir="${basedir}/src" />
    <mkdir dir="${basedir}/dist" />
    <mkdir dir="${basedir}/lib" />
  </target>
</project>
```

Avec dir, le chemin et le nom du répertoire à créer
ant -buildfile build.mkdir.xml

```
Buildfile: build.mkdir.xml
init:
    [mkdir] Created dir: /home/hernandez/teaching/TdD/05_NH_CM_ant/test/build
    [mkdir] Created dir: /home/hernandez/teaching/TdD/05_NH_CM_ant/test/src
    [mkdir] Created dir: /home/hernandez/teaching/TdD/05_NH_CM_ant/test/dist
    [mkdir] Created dir: /home/hernandez/teaching/TdD/05_NH_CM_ant/test/lib
BUILD SUCCESSFUL
Total time: 0 seconds
```

La tâche delete

<delete> supprime des fichiers ou des répertoires

build.delete.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test delete avec Ant" default="init" basedir=".">

  <target name="init">
    <delete dir="${basedir}/dist"  includeEmptyDirs="true"/>
    <delete file="${basedir}/log.txt" />
    <delete>
      <fileset dir="${basedir}/build" includes="**/*.class" />
    </delete>
     <delete>
      <fileset dir="${basedir}/src" includes="**/.svn"/>
    </delete>
  </target>
</project>
```

ant -buildfile build.delete.xml

```
Buildfile: build.delete.xml
init:
   [delete] Deleting: /home/hernandez/teaching/TdD/05_NH_CM_ant/test/log.txt
   [delete] Deleting directory /home/hernandez/teaching/TdD/05_NH_CM_ant/test/di
BUILD SUCCESSFUL
Total time: 0 seconds
```

La tâche copy

<copy> Copies a file or resource collection to a new file or directory. By default, files are only copied if the source file is newer than the destination file, or when the destination file does not exist.

build.copy.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test de copy avec ant" default="init" basedir=".">

  <!-- Definition des proprietes du projet -->
  <property name="sources.dir"    value="src"/>
  <property name="build.dir"      value="bin"/>

  <!-- Initialisation des traitements -->
  <target name="init" description="Initialisation">
    <!-- Copie des fichiers de configuration et parametrage -->
    <copy todir="${projet.build.dir}" >
      <fileset dir="${projet.sources.dir}" >
        <include name="**/*.properties"/>
        <include name="**/*.cfg.xml"/>
      </fileset>
    </copy>
  </target>
</project>
```

D'autres options sont disponibles `tofile`, `overwrite`

La tâche javac

<javac> permet la compilation de fichiers source contenus dans une arborescence de répertoires

srcdir répertoire racine de l'arborescence du répertoire contenant les sources

destdir répertoire où les résultats des compilations seront stockés

classpath classpath pour l'exécution. Il est aussi possible d'utiliser un tag fils <classpath> pour le spécifier

classpathref utilisation d'un classpath précédemment défini dans le fichier de build

fork lance la compilation dans une JVM dédiée au lieu de celle ou s'exécute Ant. défaut est false

source version des sources java 1.4, 1.5, ...

deprecation avertissements du compilateur concernant l'usage d'éléments deprecated. défaut est off

target précise la version de la plate-forme Java cible (1.1, 1.2, 1.3, 1.4, ...)

... nowarn, debug, optimize, failonerror

build.javac.xml

```
<xml version="1.0" encoding="UTF-8"?>

<project name="Test javac task" default="compile" basedir=".">
  <!-- Definition des proprietes du projet -->
  <property name="sources.dir" value="src"/>
  <property name="build.dir" value="build"/>
  <property name="lib.dir" value="lib"/>

  <!-- Definition du classpath du projet -->
  <path id="projet.classpath">
    <fileset dir="${lib.dir}">
      <include name="*.jar"/>
    </fileset>
    <pathelement location="${build.dir}" />
  </path>

  <!-- Compilation des classes du projet -->
  <target name="compile" description="Compilation des classes">
    <javac srcdir="${sources.dir}"
          destdir="${build.dir}"
  debug="on"
  optimize="off"
  deprecation="on">
      <classpath refid="projet.classpath"/>
    </javac>
  </target>
</project>
```

La tâche java

<java> permet de lancer une machine virtuelle pour exécuter une application compilée.

classname nom pleinement qualifié de la classe à exécuter

jar nom du fichier de l'application à exécuter

classpath classpath pour l'exécution.

classpathref utilisation d'un classpath précédemment défini

fork lancer l'exécution dans une JVM dédiée au lieu de celle ou l'exécute Ant

output enregistrer les sorties de la console dans un fichier

La tâche java

build.java.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<project name="Test java task" default="execute" basedir=".">
  <!-- Definition des proprietes du projet -->
  <property name="sources.dir" value="src"/>
  <property name="build.dir" value="build"/>
  <property name="lib.dir" value="lib"/>

  <!-- Definition du classpath du projet -->
  <path id="projet.classpath">
    <fileset dir="${lib.dir}">
      <include name="*.jar"/>
    </fileset>
    <pathelement location="${build.dir}" />
  </path>

  <!-- Execution de HelloWorld -->
  <target name="execute" description="Execution de HelloWorld" >
    <java classname="HelloWorld" fork="true">
      <classpath refid="projet.classpath"/>
    </java>
  </target>
</project>
```

La tâche javadoc

<javadoc> génération de la documentation au format javadoc des classes incluses dans une arborescence de répertoires

sourcepath le répertoire de base qui contient les sources dont la documentation est à générer

destdir le répertoire qui va contenir les fichiers de documentation générés

build.javadoc.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<project name="Test javadoc avec Ant" default="javadoc" basedir=".">
  <!-- ================================================= -->
  <!-- Génération de la documentation Javadoc          -->
  <!-- ================================================= -->
  <target name="javadoc">
    <mkdir dir="doc"/>
    <javadoc sourcepath="src"
             destdir="doc" >
      <fileset dir="src" defaultexcludes="yes">
        <include name="**" />
      </fileset>
    </javadoc>
  </target>
</project>
```

La tâche jar

<jar> la création d'une archive de type jar

jarfile nom du fichier .jar à créer

basedir répertoire qui contient les éléments à ajouter dans l'archive

compress sprécise si le contenu de l'archive doit être compressé ou non. Par défaut est true

manifest le fichier manifest qui sera utilisé dans l'archive

build.jar.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<project name="Test jar avec Ant" default="packaging" basedir=".">

  <!-- =================================================================== -->
  <!-- Génération de l'archive jar                                  -->
  <!-- =================================================================== -->
  <target name="packaging">
    <jar jarfile="test.jar" basedir="src" />
  </target>
</project>
```

La tâche JUnit

Installation

- Récupérer JUnit.jar `http://www.junit.org/`
- `export CLASSPATH=`*CLASSPATH* `:APPLI/JUnit/junit-4.4.jar`

Description des tâches

<junit> This task runs tests from the JUnit testing framework

<formater> print results of tests in different formats (`plain, xml`)

<test> Defines a single test class

<batchtest> Define a number of tests based on pattern matching

`http://ant.apache.org/manual/OptionalTasks/junit.html`

La tâche JUnit

build.junit.xml

```
<junit printsummary="yes" haltonfailure="yes">
  <classpath>
    <pathelement location="${build.tests}"/>
    <pathelement path="${java.class.path}"/>
  </classpath>

  <formatter type="plain"/>

  <test name="my.test.TestCase" haltonfailure="no" outfile="result">
    <formatter type="xml"/>
  </test>

  <batchtest fork="yes" todir="${reports.tests}">
    <fileset dir="${src.tests}">
      <include name="**/*Test*.java"/>
      <exclude name="**/AllTests.java"/>
    </fileset>
  </batchtest>
</junit>
```

printsummary Print one-line statistics for each testcase.
fork Run the tests in a separate VM.
haltonfailure Stop the build process if a test fails
timeout Cancel the individual tests if they don’t finish in the given time
todir Directory to write the reports to

La tâche svn

Installation

- Récupérer svnant.jar, svnClientAdapter.jar et svnjavahl.jar à partir de l'archive svnant http://subclipse.tigris.org/svnant.html
- Les mettre dans le classpath...

Description de la tâche svn

- http://subclipse.tigris.org/svnant/svn.html
- Exemple de build.xml mettant en oeuvre svnant dans l'archive récupérée ci-dessus
- Autres exemples d'utilisation subversion.open.collab.net/articles/IntegratingSubversionIntoYourAntBuild.html

La tâche svn

Un exemple de build.properties

```
# build.properties
svnant.version=1.0.0

lib.dir=lib
svnant.jar=${lib.dir}/svnant.jar
svnClientAdapter.jar=${lib.dir}/svnClientAdapter.jar
svnjavahl.jar=${lib.dir}/svnjavahl.jar

svnant.latest.url=http://subclipse.tigris.org/svn/subclipse/trunk/svnant/
svnant.this.url=http://subclipse.tigris.org/svn/subclipse/tags/svnant/${svnant.v
ersion}/

svnant.repository.user=guest
svnant.repository.passwd=""
```

build.xml (part 1/2)

```
  <!--  all properties are in build.properties -->
  <property file="build.properties" />

  <!-- path to the svnant libraries. Usually in ANT_HOME/lib -->
  <path id="project.classpath">
    <pathelement location="${svnjavahl.jar}" />
    <pathelement location="${svnant.jar}" />
    <pathelement location="${svnClientAdapter.jar}" />
```

La tâche svn

build.xml (part 2/2)

```
...
  <!-- load the svn task -->
  <taskdef resource="svntask.properties" classpathref="project.classpath"/>

  <target name="clean">
    <delete dir="src_latest"/>
    <delete dir="src_${svnant.version}"/>
  </target>

  <target name="checkoutLatest">
    <svn username="${svnant.repository.user}" password="${svnant.repository.pass
wd}">
      <checkout url="${svnant.latest.url}" revision="HEAD" destPath="src_latest"
 />
    </svn>
  </target>

  <target name="checkoutThis">
    <svn username="${svnant.repository.user}" password="${svnant.repository.pass
wd}">
      <checkout url="${svnant.this.url}" revision="HEAD" destPath="src_${svnant.
version}" />
    </svn>
  </target>
```

Définir ses propres tâches – Sommaire

Définir une tâche `ant` dans le `build.xml`
La classe implémentant cette tâche
Code de la classe implémentant la tâche

Définir une tâche ant *dans le* build.xml

Ant permet de définir ces propres tâches

```
<?xml version="1.0"?>
<project name="ExempleNotreTache" default="main" basedir=".">

<taskdef name="matache" classname="tdd.MaTacheAMoi"/>

<target name="main">
<matache message="Ant is Great !"/>
</target>
</project>
```

La classe implémentant cette tâche

- La classe implémentant cette tâche doit **étendre** `org.apache.tools.ant.Task`
- Un **accesseur en écriture** doit être défini pour chaque attribut. Il aura la forme traditionnelle en Java : `setNomAttribut`. Le type recu en paramètre de cette méthode peut être String, ou n'importe quel des types de base, Ant se chargeant des conversions.
- **Chaque sous-élément supporté par la tâche** devra de la même manière être traité par le biais de méthodes `createNomElement` **ou** `addNomElement`
- La classe doit enfin comporter **une méthode** `public void execute() throws BuildException`

Code de la classe implémentant la tâche

```
Package tdd;

import org.apache.tools.ant.BuildException;
import org.apache.tools.ant.Task;

public class MaTacheAMoi extends Task {
private String msg;

// La méthode appelée par Ant pour l'execution de la tache
public void execute() throws BuildException {
System.out.println(msg);
}

// Accesseur pour l'attribut message
public void setMessage(String msg) {
this.msg = msg;
}
}
```

Conclusion – Sommaire

ant *et Eclipse*

Documentation
`http://help.eclipse.org/help33/index.jsp?topic=/org.eclipse.platform.doc.user/`

- puis `gettingStarted/qs-81_basics.htm`
- ou `concepts/concepts-antsupport.htm`

En bref...

Créer un projet à partir d'un `ant buildfile` *ou en ajouter un*

- soit `File > New > Java Project > Create a new project java from an existing ant buildfile`
- soit `File > New > File`

Exécuter

- dans la vue `Package`, bouton droit sur le fichier build.xml sélectionné
- ou bien Icone *Run avec la malette* de la barre de menu horizontale
- ou bien dans la vue *Outline*, bouton droit sur une des cibles

Puis `Run as > Ant Build`

Sujet du TP

Construire pour votre projet un `build.xml` contenant les cibles :

- *init* Initialisation, définition des propriétés
- *prepare* Préparation (création des répertoires par exemple)
- *clean* Suppression de tout ce que le processus de construction peut produire
- *compile* Compilation des sources de l'application
- *compile-test* Compilation des sources de test
- *test* Exécution des tests unitaires
- *javadoc* Création de la Javadoc
- *dist* Création d'une archive des sources de la distribution
- *deploy* Déploiement de l'application sur le serveur cible
- *properties* Affichage des valeurs de la configuration
- *usage* Affichage des cibles disponibles
- *clean* (nettoie les .class produits et les répertoires inutiles)
- *checkout/commit/update/diff/patch/tag* (versionning avec svn)

Conclusion

Synthèse

- multi-plate-forme
- configurable grâce à un fichier XML
- open-source
- extensible

Perspective

- actuellement la version 1.7 depuis octobre 2006 (possibilité de problèmes de compatibilité suivant la version installée)
- `make` le passé et `maven`, le futur ?

Bibliographie

Gestion de Version

- Site officiel http ://ant.apache.org/
- Manuel d'utilisateur http ://ant.apache.org/manual/index.html
- Tutoriels http://ant.apache.org/manual/tutorials.html (Hello World with Ant, Writing Tasks, Tasks using Properties, Filesets & Paths)
- Ant et Eclipse
  http ://help.eclipse.org/help33/index.jsp?topic=/org.eclipse.platform.doc.user/
  puis gettingStarted/qs-81_basics.htm ou
  concepts/concepts-antsupport.htm
- Manuels non-officiel et non-complet de ant et maven (en français)
  http ://www.jmdoudoux.fr/java/dej/index.htm
- Ant et JUnit
  http://ant.apache.org/manual/OptionalTasks/junit.html
- Ant et svn http://subclipse.tigris.org/svnant.html